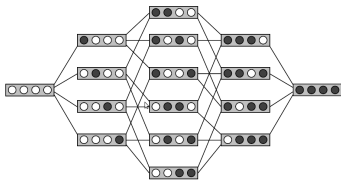


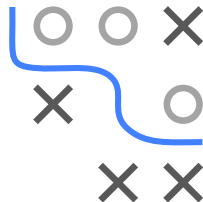
# Supervised Learning

## Wrapper methods



### Learning goals

- Understand how wrapper methods work
- Understand how they could help in feature selection
- Know their advantages and disadvantages



# INTRODUCTION

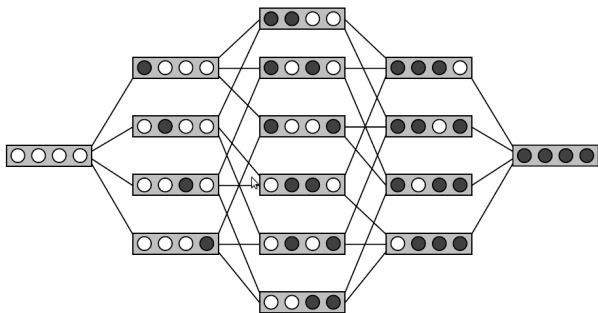
- Wrapper methods emerged from the idea that different sets of features can be optimal for different learners.
- Use the learner itself to assess the quality of the feature sets.
- Evaluation on a test set or resampling techniques are used.
- A wrapper is nothing else than a discrete search strategy for  $S$ , where the test error of a learner as a function of  $S$  is now the objective criterion.



# INTRODUCTION

Wrappers have the following components:

- A set of starting values
- Operators to create new points out of the given ones
- A termination criterion

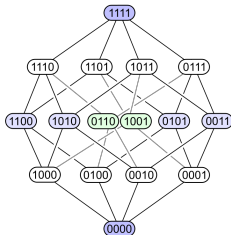


# OBJECTIVE FUNCTION

Given  $p$  features, the **best-subset selection problem** is to find a subset  $S \subseteq \{1, \dots, p\}$  optimizing objective  $\Psi : \Omega \rightarrow \mathbb{R}$ :

$$S^* \in \arg \min_{S \in \Omega} \{\Psi(S)\}$$

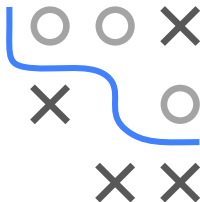
- $\Omega$  = search space of all feature subsets  $S \subseteq \{1, \dots, p\}$ . Usually we encode this by bit vectors, i.e.,  $\Omega = \{0, 1\}^p$  (1 = feat. selected)
- Objective  $\Psi$  can be different functions, e.g., AIC/BIC for LM or cross-validated performance of a learner.



Hasse diagram (source: Wikipedia)

# HOW DIFFICULT IS BEST-SUBSET SELECTION?

- Size of search space =  $2^p$ , i.e., grows exponentially in  $p$  as it is the power set of  $\{1, \dots, p\}$ .
- Finding best subset is discrete combinatorial optimization problem also known as  $L_0$  regularization.
- It can be shown that this problem unfortunately can not be solved efficiently in general (NP hard; see, e.g., [Natarajan, 1995](#))
- We can avoid having to search the entire space by employing efficient search strategies, moving through the search space in a smart way that finds performant feature subsets.



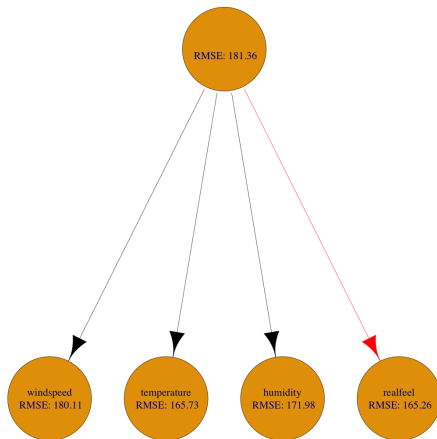
# GREEDY FORWARD SEARCH

- Let  $S \subset \{1, \dots, p\}$ , where  $\{1, \dots, p\}$  is an index set of all features.
- Start with the empty feature set  $S = \emptyset$ .
- For a given set  $S$ , generate all  $S_j = S \cup \{j\}$  with  $j \notin S$ .
- Evaluate the classifier on all  $S_j$  and use the best  $S_j$ .
- Iterate over this procedure.
- Terminate if:
  - the performance measure doesn't improve enough.
  - a maximum number of features is used.
  - a given performance value is reached.

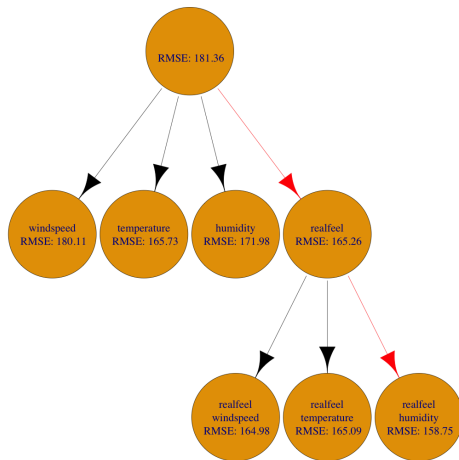


# GREEDY FORWARD SEARCH

Example for greedy forward search on bike sharing data:

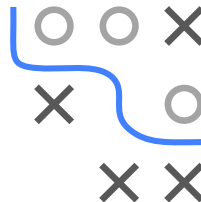
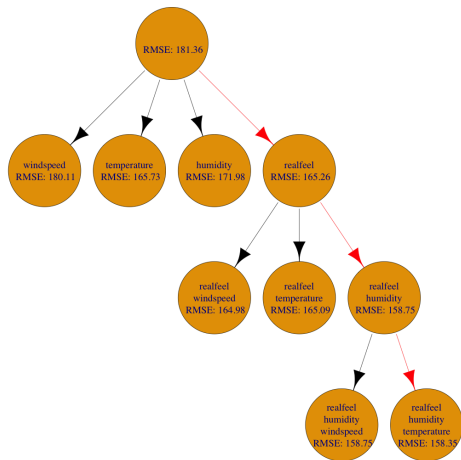


# GREEDY FORWARD SEARCH

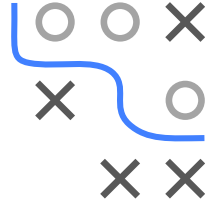
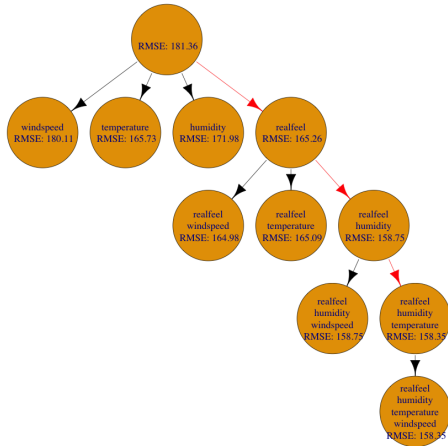




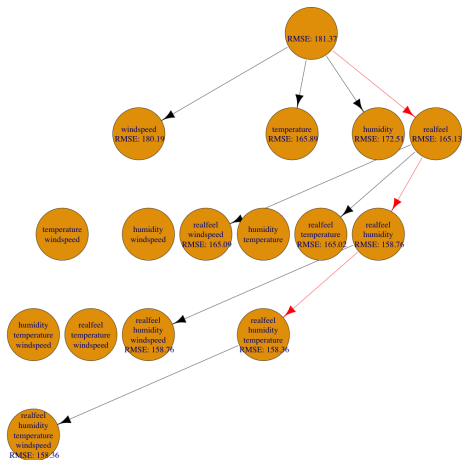
# GREEDY FORWARD SEARCH



# GREEDY FORWARD SEARCH

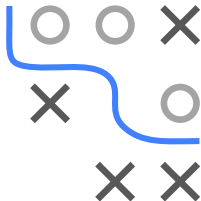


## GREEDY FORWARD SEARCH

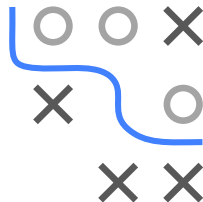
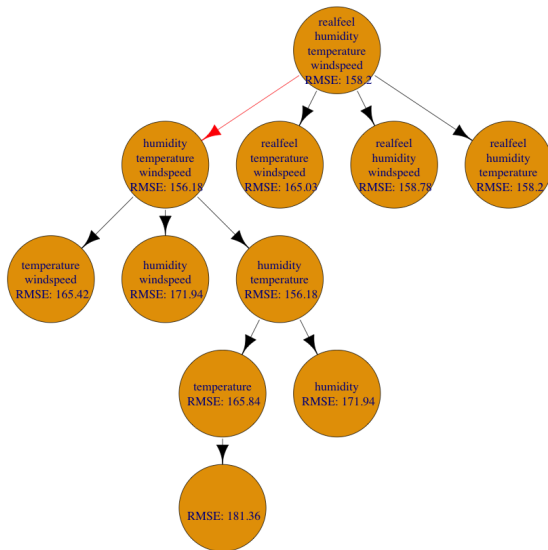


# GREEDY BACKWARD SEARCH

- Start with the full index set of features  $S = \{1, \dots, p\}$ .
- For a given set  $S$  generate all  $S_j = S \setminus \{j\}$  with  $j \in S$ .
- Evaluate the classifier on all  $S_j$  and use the best  $S_j$ .
- Iterate over this procedure.
- Terminate if:
  - the performance drops drastically, or
  - a given performance value is undershot.



# GREEDY BACKWARD SEARCH



# EXTENSIONS

- Eliminate or add several features at once to increase speed.
- Allow alternating forward and backward search.
- Randomly create candidate feature sets in each iteration.
- Continue search based on the set of features where an improvement is present.
- Use improvements of earlier iterations.



# EXTENSIONS

---

**Algorithm** A simple 1+1 genetic algorithm

---

- 1: Start with a random set of features  $S$  (bit vector  $b$ ).
  - 2: **repeat**
  - 3:     Flip a couple of bits in  $b$  with probability  $p$ .
  - 4:     Generate set  $S'$  and bit vector  $b'$ .
  - 5:     Measure the classifier's performance on  $S'$ .
  - 6:     If  $S'$  performs better than  $S$ , update  $S \leftarrow S'$ , otherwise  $S \leftarrow S$ .
  - 7: **until** One of the following conditions is met:
    - A given performance value is reached.
    - Budget is exhausted.
- 



# WRAPPERS

## Advantages:

- Can be combined with every learner.
- Can be combined with every performance measure.
- Optimizes the desired criterion directly.



## Disadvantages:

- Evaluating the target function is expensive.
- Does not scale well if number of features becomes large.
- Does not use much structure or available information from our model.